APPARATUS AND METHODS FOR MAXIMIZING SERVICE-LEVEL-AGREEMENT PROFITS

1. Technical Field:

The present invention is directed to an improved distributed computer system. More particularly, the present invention is directed to apparatus and methods for maximizing service-level-agreement (SLA) profits.

2. Description of Related Art:

As the exponential growth in Internet usage continues, much of which is fueled by the growth and requirements of different aspects of electronic business (e-business), there is an increasing need to provide Quality of Service (QoS) performance guarantees across a wide range of high-volume commercial Web site environments. A fundamental characteristic of these commercial environments is the diverse set of services provided to support customer requirements. Each of these services have different levels of importance to both the service providers and their clients. To this end, Service Level Agreements (SLAs) are established between service providers and their clients so that different QoS requirements can be satisfied. This gives rise to the definition of different classes of services. Once a SLA is in effect, the service providers must make appropriate resource management decisions to accommodate these SLA service classes.

One such environment in which SLAs are of increasing importance is in Web server farms. Web server farms are becoming a major means by which Web sites are hosted. The basic architecture of a Web server farm is a cluster of Web servers that allow various Web sites to share the resources of the farm, i.e. processor resources, disk storage, communication bandwidth, and the like. In this way, a Web server farm supplier may host Web sites for a plurality of different clients.

In managing the resources of the Web server farm, traditional resource management mechanisms attempt to optimize conventional performance metrics such as mean response time and throughput. However, merely optimizing performance metrics such as mean response time

and throughput does not take into consideration tradeoffs that may be made in view of meeting or not meeting the SLAs being managed. In other words, merely optimizing performance metrics does not provide an indication of the amount of revenue generated or lost due to meeting or not meeting the service level agreements.

Thus, it would be beneficial to have an apparatus and method for managing system resources under service level agreements based on revenue metrics rather than strictly using conventional performance metrics in order to maximize the amount of profit generated under the SLAs.

SUMMARY OF THE INVENTION

The present invention provides apparatus and methods for maximizing service-level-agreement (SLA) profits. The apparatus and methods consist of formulating SLA profit maximization as a network flow model with a separable set of concave cost functions at the servers of a Web server farm. The SLA classes are taken into account with regard to constraints and cost function where the delay constraints are specified as the tails of the corresponding response-time distributions. This formulation simultaneously yields both optimal load balancing and server scheduling parameters under two classes of server scheduling policies, Generalized Processor Sharing (GPS) and Preemptive Priority Scheduling (PPS). For the GPS case, a pair of optimization problems are iteratively solved in order to find the optimal parameters that assign traffic to servers and server capacity to classes of requests. For the PPS case, the optimization problems are iteratively solved for each of the priority classes, and an optimal priority hierarchy is obtained.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is an exemplary block diagram illustrating a network data processing system according to one embodiment of the present invention;

Figure 2 is an exemplary block diagram illustrating a server device according to one embodiment of the present invention;

Figure 3 is an exemplary block diagram illustrating a client device according to one embodiment of the present invention;

Figure 4 is an exemplary diagram of a Web server farm in accordance with the present invention;

Figure 5 is an exemplary diagram illustrating this Web server farm model according to the present invention;

Figures 6A and **6B** illustrate a queuing network in accordance with the present invention;

Figure 7 is an exemplary diagram of a network flow model in accordance with the present invention;

Figure 8 is a flowchart outlining an exemplary operation of the present invention in a GPS scheduling environment; and

Figure 9 is a flowchart outlining an exemplary operation of the present invention in a PPS scheduling environment.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

As mentioned above, the present invention provides a mechanism by which profits generated by satisfying SLAs are maximized. The present invention may be implemented in any distributed computing system, a stand-alone computing system, or any system in which a cost model is utilized to characterize revenue generation based on service level agreements. Because the present invention may be implemented in many different computing environments, a brief discussion of a distributed network, server computing device, client computing device, and the like, will now be provided with regard to **Figures 1-3** in order to provide an context for the exemplary embodiments to follow. Although a preferred implementation in Web server farms will be described, those skilled in the art will recognize and appreciate that the present invention is significantly more general purpose and is not limited to use with Web server farms.

With reference now to the figures, **Figure 1** depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented. Network data processing system **100** is a network of computers in which the present invention may be implemented. Network data processing system **100** contains a network **102**, which is the medium used to provide communications links between various devices and computers connected together within network data processing system **100**. Network **102** may include connections, such as wire, wireless communication links, or fiber optic cables.

In the depicted example, a server **104** is connected to network **102** along with storage unit **106**. In addition, clients **108**, **110**, and **112** also are connected to network **102**. These clients **108**, **110**, and **112** may be, for example, personal computers or network computers. In the depicted example, server **104** provides data, such as boot files, operating system images, and applications to clients **108-112**. Clients **108**, **110**, and **112** are clients to server **104**. Network data processing system **100** may include additional servers, clients, and other devices not shown.

In the depicted example, network data processing system **100** is the Internet with network **102** representing a worldwide collection of networks and gateways that use the TCP/IP suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, consisting of thousands of commercial, government, educational and other computer systems that route data

and messages. Of course, network data processing system **100** also may be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). **Figure 1** is intended as an example, and not as an architectural limitation for the present invention.

In addition to the above, the distributed data processing system **100** may further include a Web server farm **125** which may host one or more Web sites **126-129** for one or more Web site clients, e.g. electronic businesses or the like. For example, the Web server farm **125** may host a Web site for "Uncle Bob's Fishing Hole" through which customers may order fishing equipment, a Web site for "Hot Rocks Jewelry" through which customers may purchase jewelry at wholesale prices, and a Web site for "Wheeled Wonders" where customers may purchase bicycles and bicycle related items.

A user of a client device, such as client device **108** may log onto a Web site hosted by the Web server farm **125** by entering the URL associated with the Web site into a Web browser application on the client device **108**. The user of the client device 108 may then navigate the Web site using his/her Web browser application, selecting items for purchase, providing personal information for billing purposes, and the like.

With the present invention, the Web site clients, e.g. the electronic businesses, establish service level agreements with the Web server farm **125** provider regarding various classes of service to be provided by the Web server farm **125**. For example, a service level agreement may indicate that a browsing client device is to be provided a first level of service, a client device having an electronic shopping cart with an item therein is provided a second level of service, and a client device that is engaged in a "check out" transaction is given a third level of service. Based on this service level agreement, resources of the Web server farm are allocated to the Web sites of the Web site clients to handle transactions with client devices. The present invention is directed to managing the allocation of these resources under the service level agreements in order to maximize the profits obtained under the service level agreements, as will be described in greater detail hereafter.

Referring to **Figure 2**, a block diagram of a data processing system that may be implemented as a server, such as server **104** or a server in the Web server farm **125** in **Figure 1**, is depicted in accordance with a preferred embodiment of the present invention. Data processing

system **200** may be a symmetric multiprocessor (SMP) system including a plurality of processors **202** and **204** connected to system bus **206**. Alternatively, a single processor system may be employed. Also connected to system bus **206** is memory controller/cache **208**, which provides an interface to local memory **209**. I/O bus bridge **210** is connected to system bus **206** and provides an interface to I/O bus **212**. Memory controller/cache **208** and I/O bus bridge **210** may be integrated as depicted.

Peripheral component interconnect (PCI) bus bridge **214** connected to I/O bus **212** provides an interface to PCI local bus **216**. A number of modems may be connected to PCI bus **216**. Typical PCI bus implementations will support four PCI expansion slots or add-in connectors. Communications links to network computers **108-112** in **Figure 1** may be provided through modem **218** and network adapter **220** connected to PCI local bus **216** through add-in boards.

Additional PCI bus bridges **222** and **224** provide interfaces for additional PCI buses **226** and **228**, from which additional modems or network adapters may be supported. In this manner, data processing system **200** allows connections to multiple network computers. A memory-mapped graphics adapter **230** and hard disk **232** may also be connected to I/O bus **212** as depicted, either directly or indirectly.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 2** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

The data processing system depicted in **Figure 2** may be, for example, an IBM RISC/System 6000 system, a product of International Business Machines Corporation in Armonk, New York, running the Advanced Interactive Executive (AIX) operating system.

With reference now to **Figure 3**, a block diagram illustrating a data processing system is depicted in which the present invention may be implemented. Data processing system **300** is an example of a client computer. Data processing system **300** employs a peripheral component interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used. Processor **302** and main memory **304** are connected to PCI

local bus **306** through PCI bridge **308**. PCI bridge **308** also may include an integrated memory controller and cache memory for processor **302**. Additional connections to PCI local bus **306** may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter **310**, SCSI host bus adapter **312**, and expansion bus interface **314** are connected to PCI local bus **306** by direct component connection. In contrast, audio adapter **316**, graphics adapter **318**, and audio/video adapter **319** are connected to PCI local bus **306** by add-in boards inserted into expansion slots. Expansion bus interface **314** provides a connection for a keyboard and mouse adapter **320**, modem **322**, and additional memory **324**. Small computer system interface (SCSI) host bus adapter **312** provides a connection for hard disk drive **326**, tape drive **328**, and CD-ROM drive **330**. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors.

An operating system runs on processor **302** and is used to coordinate and provide control of various components within data processing system **300** in **Figure 3**. The operating system may be a commercially available operating system, such as Windows 2000, which is available from Microsoft Corporation. An object oriented programming system such as Java may run in conjunction with the operating system and provide calls to the operating system from Java programs or applications executing on data processing system **300**. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented operating system, and applications or programs are located on storage devices, such as hard disk drive **326**, and may be loaded into main memory **304** for execution by processor **302.**

Those of ordinary skill in the art will appreciate that the hardware in **Figure 3** may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure 3**. Also, the processes of the present invention may be applied to a multiprocessor data processing system.

As another example, data processing system **300** may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data processing system **300** comprises some type of network communication interface. As a further example, data processing system **300** may be a Personal Digital Assistant (PDA) device,

which is configured with ROM and/or flash ROM in order to provide non-volatile memory for storing operating system files and/or user-generated data.

The depicted example in **Figure 3** and above-described examples are not meant to imply architectural limitations. For example, data processing system **300** also may be a notebook computer or hand held computer in addition to taking the form of a PDA. Data processing system **300** also may be a kiosk or a Web appliance.

The present invention provides a mechanism by which resources are managed so as to maximize the profit generated by satisfying service level agreements. The present invention will be described with regard to a Web server farm, however the invention is not limited to such. As mentioned above, the present invention may be implemented in a server, client device, stand-alone computing system, Web server farm, or the like.

With the preferred embodiment of the present invention, as shown in **Figure 4**, a Web server farm **400** is represented by a distributed data processing system consisting of M heterogeneous servers that independently execute K classes of request streams, where each request is destined for one of N different Web client Web sites. As shown in **Figure 4**, the Web server farm **400** includes a request dispatcher **410** coupled to plurality of servers **420-432**. The request dispatcher **410** receives requests via the network **102** destined for a Web site supported by the Web server farm **400**. The request dispatcher **410** receives these requests, determines an appropriate server to handle the request, and reroutes the request to the identified server. The request dispatcher **410** also serves as an interface for outgoing traffic from the Web server farm **400** to the network **102**.

Every Web site supported by the Web server farm **400** has one or more classes of requests which may or may not have service level agreement (SLA) requirements. The requests of each class for each Web site may be served by a subset of the servers **420-432** comprising the Web server farm **400**. Further, each server **420-432** can serve requests from a subset of the different class-Web site pairs.

To accommodate any and all restrictions that may exist in the possible assignments of class-Web site pairs to servers (e.g., technical, business, etc.), these possible assignments are given via a general mechanism. Specifically, if *A(i,j,k)* is the indicator function for these assignments, *A(i,j,k)* takes on the value 0 or 1, where 1 indicates that class k requests destined for

Web site j can be served by server i and 0 indicates they cannot. Thus, A(i,j,k) simply defines the set of class-Web site requests that can be served by a given server of the Web server farm.

The present invention provides a mechanism for controlling the routing decisions between each request and each server eligible to serve such request. More precisely, the present invention determines an optimal proportion of traffic of different classes to different Web sites to be routed to each of the servers. Thus, the present invention determines which requests are actually served by which servers in order to maximize profits generated under SLAs.

Web clients use the resources of the Web server farm **400** through their navigation behavior on the hosted Web sites. This navigational behavior is characterized by Web sessions consisting of a sequence of alternating actions. A typical Web client scenario might consist of several browse requests, possibility followed by an add-to-shopping cart request or buy transaction request, in an iterative manner. Between requests, there may be client device-based delays, which can represent user "think times," fixed time intervals generated by a computer (e.g., Web crawlers or the like), Web browser application delays (e.g., upon requesting embedded images), and the like. This sequence can be finite or infinite, with the latter case corresponding to Web crawler activities. For a single session, the requests may belong to different classes and the think times may be of different types.

The present invention is premised on the concept that revenue may be generated each time a request is served in a manner that satisfies the corresponding service level agreement. Likewise, a penalty may be paid each time a request is not served in a manner that satisfies the corresponding service level agreement. The only exception to this premise is "best efforts" requirements in service level agreements which, in the present invention, have a flat rate pricing policy with zero penalty. Thus, the profit generated by hosting a particular Web site on a Web server farm is obtained by subtracting the penalties from the revenue generated. The present invention is directed to maximizing this profit by efficiently managing the Web server farm resources.

Web Server Farm Model

With the present invention, the Web server farm is modeled by a multiclass queuing network composed of a set of M single-server multiclass queues and a set of NxKxK queues. The former represents a collection of heterogeneous Web servers and the latter represents the client device-based delays (or "think times") between the service completion of one request and the arrival of a subsequent request within a Web session. **Figure 5** is an exemplary diagram illustrating this Web server farm model. For convenience, the servers of the first set, i.e. the Web servers, are indexed by i, i=1,...,M and those of the second set (delay servers) are indexed by (j, k, k'), the Web client sites by j, j=1,...,N, and the request classes by k, k=1,...,K.

For those M single-server multiclass queues representing the Web servers, it is assumed, for simplicity, that each server can accommodate all classes of requests, however the invention is not limited to such an assumption. Rather, the present invention may be implemented in a Web server farm in which each server may accommodate a different set of classes of requests, for example. The service requirements of class k requests at server i follow an arbitrary distribution with mean $l_{i,k}^{-1}$. The capacity of server i is denoted by C_i.

The present invention may make use of either a Generalized Processor Sharing (GPS) or Preemptive Priority Scheduling (PPS) scheduling policy to control the allocation of resources across the different service classes on each server. Under GPS, each class of service is assigned a coefficient, referred to as a GPS assignment, such that the server capacity is shared among the classes in proportion to their GPS assignments. Under PPS, scheduling is based on relative priorities, e.g. class 1 requests have a highest priority, class 2 requests have a lower priority than class 1, and so on.

In the case of GPS, the GPS assignment to class k on server i is denoted by $f_{i,k}$ with the sum of $f_{i,k}$ over the range of k=1 to k=K being 1. Thus, at any time t, the server capacity devoted to class k requests, if any, is $f_{i,k}C/S_{k' \in Ki(t)} f_{i,k}$, where $K_i(t)$ is the set of classes with backlogged requests on server i at time t. Requests within each class are executed either in a First-Come-First Served (FCFS) manner or in a processor sharing (PS) manner. User transactions from a client device destined for a Web site j that begin with a class k request, arrive to the distributed

data processing system, i.e. the Web server farm, from an exogenous source with rate $k_k^{(j)}$. Upon completion of a class k request, the corresponding Web site j user transaction either returns as a class k' request with probability $p^{(j)}_{k,k'}$ following a delay at a queue having mean $(d^{(j)}_{k,k'})^{-1}$, or completes with probability $1-S^{K}_{l=1} p^{(j)}_{k,l}$. The matrix $P^{(j)}=[p^{(j)}_{k,k'}]$ is the corresponding request feedback matrix for Web site j which is substochastic and has dimension KxK. This transition probability matrix $P^{(j)}$ defines how each type of Web site j user transaction flows through the queuing network as a sequence of server requests and client think times. Thus, this matrix may be used to accurately reflect the inter-request correlations resulting from client-server interactions. The client device think times can have arbitrary distributions, depending on the Web site and the current and future classes. These think times are used in the model to capture the complete range of causes of delays between the requests of a user session including computer delays (e.g., Web crawlers and Web browsers) and human delays.

$L_k^{(j)}$ denotes the rate of aggregate arrivals of Web site j, class k requests to the set of servers of the Web server farm. The rate of aggregate arrivals may be determined based on the exogenous arrival rates and the transition probabilities as follows:

$$L_k^{(j)} = \sum_{k'=1}^{K} L_k^{(j)} p^{(j)}_{k,k'} + k_k^{(j)}, j=1,...,N, k=1,...,K \qquad (1)$$

While the above models uses a Markovian description of user navigational behavior, the present invention is not limited to such. Furthermore, by increasing the number of classes and thus, the dimensions of the transition probability matrix, any arbitrary navigational behavior with particular sequences of request classes may be modeled. In such cases, many of the entries of the transition probability matrix $P^{(j)}$ will be 0 or 1. In so doing, any arbitrary distribution of the number of requests within a session may be approximated.

Cost Model

As mentioned above, the present invention is directed to maximizing the profit generated by hosting a Web site on a Web server farm. Thus, a cost model is utilized to represent the costs involved in hosting the Web site. In this cost model, $k_{i,k}^{(j)}$ is used to denote the rate of class k

requests destined for Web site j that are assigned to server *i* by the control policy of the present invention. The scheduling discipline, either GPS or PPS, at each single-server multiclass queue determines the execution ordering across the request classes.

The cost model is based on the premise that profit is gained for each request that is processed in accordance with its per-class service level agreement. A penalty is paid for each request that is not processed in accordance with its per-class service level agreement. More precisely, assume T_k is a generic random variable for the class k response time process, across all servers and all Web sites. Associated with each request class k is a SLA of the form:

$$P[T_k>z_k][a_k \qquad (2)$$

where z_k is a delay constraint and a_k is a tail distribution objective. In other words, the class *k* SLA requires that the response times of requests of class *k* across all Web sites must be less than or equal to z_k at least $(1-a_k)*100$ percent of the time in order to avoid SLA violation penalties. Thus, the cost model is based on incurring a profit P_k^+ for each class *k* request having a response time of at most z_k (i.e. satisfying the tail distribution objective) and incurring a penalty P_k^- for each class *k* request which has a response time exceeding z_k (i.e. fails the tail distribution objective).

One request class is assumed to not have an SLA and is instead served on a best effort basis. The cost model for each best effort class *k* is based on the assumption that a fixed profit P_k^+ is gained for the entire class, independent of the number of class *k* requests executed. For simplicity, it will be assumed that there is only one best effort class, namely class *K*, however it is straightforward to extend the present invention to any number of best effort classes.

SLA Profit Maximization: GPS Case

Resource management with the goal of maximizing the profit gained in hosting Web sites under SLA constraints will now be considered. As previously mentioned, the foregoing Web server farm model and cost models will be considered under two different local scheduling

policies for allocating server capacity among classes of requests assigned to each server. These two policies are GPS and PPS, as previously described. The GPS policy will be described first.

In the GPS policy case, the aim is to find the optimal traffic assignments $k_{i,k}^{(j)}$ and GPS assignments $f_{i,k}$ that maximize the profit, given the class-Web site assignments $A(i,j,k)$ and the exogeneous arrival rates $k_k^{(j)}$ which yields the aggregate arrival rates $L_k^{(j)}$ through equation (1). Here $k_{i,k}^{(j)}$ denotes the rate of class k, Web site j requests assigned to server i and $f_{i,k}$ denotes the GPS assignment for class k at server i.

Routing decisions at the request dispatcher **410** are considered to be random, i.e. a class k request for site j is routed to server i with probability $k_{i,k}^{(j)}/S^{M}_{i'=1}k_{i',k}^{(j)}$, independent of the past and future routing decisions. When other routing mechanisms are used, such as weighted round robin, the resource management solutions of the present invention may be used for setting the parameters of these mechanisms (e.g., the weights of the weighted round robin), thus yielding suboptimal solutions.

With the present invention, the queuing network model described above is first decomposed into separate queuing systems. Then, the optimization problem is formulated as the sum of costs of these queuing systems. Finally, the optimization problem is solved. Thus, by summing the profits and penalties of each queuing system and then summing the profits and penalties over all of the queuing systems for a particular class k request to a Web site i, a cost function may be generated for maintaining the Web site on the Web server farm. By maximizing the profit in this cost function, resource management may be controlled in such a way as to maximize the profit of maintaining the Web site under the service level agreement.

In formulating the optimization problem as a sum of the costs of the individual queuing systems, only servers 1,...,M need be considered and these queuing systems may be considered to have arrivals of rate $k_{i,k}\,hS^{N}_{j=1}k_{i,k}^{(j)}$ for each of the classes k=1,...,K on each of the servers i=1,...,M. The corresponding queuing network is illustrated in **Figures 6A** and **6B**.

These queuing systems are analyzed by deriving tail distributions of sojourn times in these queues in view of the SLA constraints. Bounding techniques are utilized to decompose the multiclass queue associated with server i into multiple single-server queues with capacity $f_{i,k}C_i$. The resulting performance characteristics (i.e. sojourn times) are upper bounds on those in the original systems.

For simplicity of the analysis and tractability of the optimization, the GPS assignments are assumed to satisfy $k_{i,k} < l_{i,k} f_{i,k} C_i$. It then follows from standard queuing theory that the equation (2) can be bound on the left-hand side by

$$P[T_k > z_k] \leq \exp(-(l_{i,k} f_{i,k} C_i - k_{i,k}) z_k), \; k = 1, \ldots, K-1 \qquad (3)$$

Hence, the SLA constraint is satisfied when

$$P[T_k > z_k] \leq \exp(-(l_{i,k} f_{i,k} C_i - k_{i,k}) z_k) \leq a_k, \; k = 1, \ldots, K-1 \qquad (4)$$

Next, the optimization problem is divided into separate formulations for the SLA-based classes and the best effort class. As a result of equation (4), the formulation of the SLA classes is given by:

$$\text{Max} \quad \sum_{i=1}^{M} \sum_{k=1}^{K-1} P_k^{+} k_{i,k} - (P_k^{+} + P_k^{-}) k_{i,k} \exp(-(l_{i,k} f_{i,k} C_i - k_{i,k}) z_k) \qquad (5)$$

$$\text{s.t. } k_{i,k} \leq \ln(a_k z_k)/z_k + l_{i,k} f_{i,k} C_i, \; i = 1, \ldots, M, \; k = 1, \ldots, K-1;$$

$$\sum_{j=1}^{N} k_{i,k}^{(j)} = k_{i,k}, \; i = 1, \ldots, M, \; k = 1, \ldots, K-1;$$

$$\sum_{i=1}^{M} k_{i,k}^{(j)} = L_k^{(j)}, \; j = 1, \ldots, N, \; k = 1, \ldots, K-1;$$

$$k_{i,k}^{(j)} = 0, \text{ if } A(i,j,k) = 0, \; k = 1, \ldots, K-1; \; \forall i,j$$

$$k_{i,k}^{(j)} \geq 0, \text{ if } A(i,j,k) = 1, \; k = 1, \ldots, K-1; \; \forall i,j$$

$$\sum_{k=1}^{K-1} = f_{i,k} \leq 1, \; i = 1, \ldots, M.$$

where z_k is a scaling factor for the SLA constraint a_k, and Ci is the capacity of server i. Here, the $k_{i,k}^{(j)}$ and $f_{i,k}$ are the decision variables sought and P_k^{+}, P_k^{-}, $C_i, k_k^{(j)}, z_k, a_k, z_k$, and $l_{i,k}$ are input

parameters. By allowing z_k to go to infinity for any class k, the cost model makes it possible to include the objective of maximizing the throughput for class k.

The formulation for the optimal control problem for the best efforts classes attempts to minimize the weighted sum of the expected response time for class K requests over all servers, and yields:

$$\text{Min} \quad \sum_{i=1}^{M} n_{i,K}\left(\left(S^{K}_{k=1} k_{i,k} b_{i,k}^{(2)}\right)/(2(1-q_{i,K-1}^{+})(1-q_{i,K}^{+})) + b_{i,K}/(1-q_{i,K-1}^{+})\right) \qquad (6)$$

$$\text{s.t. } k_{i,k}[\ \bar{l}_{i,K} C_i,\ i=1,\ldots,M;$$

$$\bar{C}_i = Ci - \sum_{k=1}^{K-1} k_{i,k}/l_{i,k},\ i=1,\ldots,M;$$

$$\sum_{i=1}^{M} k_{i,K}^{(j)} = L_K^{(i)},\ j=1,\ldots,N;$$

$$k_{i,K}^{(j)}=0, \text{ if } A(i,j,K)=0,\ i=1,\ldots,M, j=1,\ldots,N;$$
$$k_{i,K}^{(j)}\text{m0, if } A(i,j,K)=1,\ i=1,\ldots,M, j=1,\ldots,N.$$

where $n_{i,K}$ is the weighting factor for the expected response time of the best effort class K on server i, $b_{i,k}$, $b_{i,k}^{(2)}$ are the first two moments of the service times ($b_{i,k} = l_{i,k}/C_i$), and $q_{i,k}^{+}$ is the total load of classes 1,...,k:

$$q_{i,k}^{+} = \sum_{k'=1}^{k} q_{i,k'} \text{ h } \sum_{k'=1}^{k} k_{i,k'} b_{i,k'}/C_i \qquad (7)$$

Here, $k_{i,k}^{(j)}$ are the decision variable that are sought and the remaining variables are input parameters.

The expression of the response time in the above cost function comes from the queuing results on preemptive M/G/1 queues, which are describe in, for example, H. Takasi, Queuing Analysis, vol. 1, North-Holland, Amsterdam, 1991, pages 343-347, which is hereby incorporated by reference. The use of these queuing results is valid since the SLA classes are assigned the total capacity. The GPS assignment for best effort class requests is 0, which results in a priority

scheme between SLA classes and the best effort class. Furthermore, owing to the product-form solution, a Poisson model may be used for higher priority class requests.

The weights $n_{i,K}$ are included in the formulation as they may be greater use when there are multiple best efforts classes, e.g., classes K through K'. As a simple example, the weights $n_{i,K}$ may be set to $P_k^+/(P_K^+ + ... + P_{K+K'}^+)$ in this case.

In the above formulation of equation (5), the scaling factors $z_k > 1$ are used to generalize the optimization problem. Several practical considerations motivate the use of such scaling factors. Observe first that the use of most optimization algorithms requires that the cost functions be explicit and exhibit certain properties, e.g., differentiability and convexity/concavity. However, for scheduling policies like GPS, the resulting queuing model is usually very difficult to analyze and bounding or approximation techniques have to be used in order to obtain tail distributions. Such an approach results in a bound for the GPS scheduling policy. Thus, the use of the scaling factors allows this bias to be corrected.

Secondly, queuing models are only mathematical abstractions of the real system. Users of such queuing models usually have to be pessimistic in the setting of model parameters. Once again, the scaling factors $z_k > 1$ may be useful to bridge the gap between the queuing theoretic analysis and the real system. Furthermore, the scaling factors $z_k > 1$ make it possible for the hosting company to violate the SLA to a controlled degree in an attempt to increase profits under equation (5), whereas the hosting company will strictly follow the predefined SLA whenever $z_k = 1$.

There are two sets of decision variables in the formulation of the optimal control problem shown in equation (5), namely, $k_{i,k}^{(j)}$ and $f_{i,k}$, where the latter variables control the local GPS policy at each server. To address this problem, two subproblems are considered in an iterative manner using the same formulation with appropriately modified constraints to solve for the decision variables $k_{i,k}^{(j)}$ and $f_{i,k}$. Specifically, the following equations are iteratively solved to solve for the decision variables:

$$\text{Max} \quad \sum_{i=1}^{M} \sum_{k=1}^{K-1} P_k^+ k_{i,k} - (P_k^+ + P_k^-) k_{i,k} \exp(-(l_{i,k} f_{i,k} C_i - k_{i,k}) z_k) \qquad (8)$$

$$\text{s.t. } k_{i,k} [\, ln(a_k z_k)/z_k + l_{i,k} f_{i,k} C_i,\ i=1,...,M,\ k=1,...,K-1;$$

$$\sum_{j=1}^{N} k_{i,k}^{(j)} = k_{i,k},\ i=1,\ldots,M,\ k=1,\ldots,K\text{-}1;$$

$$\sum_{i=1}^{M} k_{i,k}^{(j)} = L_k^{(j)},\ j=1,\ldots,N,\ k=1,\ldots,K\text{-}1;$$

$k_{i,k}^{(j)}=0$, if $A(i,j,k)=0$, $k=1,\ldots,K\text{-}1$; ¼$_{i,j}$;
$k_{i,k}^{(j)}$m0, if $A(i,j,k)=1$, $k=1,\ldots,K\text{-}1$; ¼$_{i,j}$.

$$\text{Max} \quad \sum_{i=1}^{M}\sum_{k=1}^{K-1} P_k^{+}k_{i,k}-(P_k^{+}+P_k^{-})k_{i,k}\exp(-(l_{i,k}f_{i,k}C_i-k_{i,k})z_k) \qquad (9)$$

s.t. $f_{i,k}$m $k_{i,k}/l_{i,k}C_i$ - $ln(a_kz_k)/z_kl_{i,k}C_i$, $i=1,\ldots,M$,
$k=1,\ldots,K\text{-}1$;

Equation (8) is an example of a network flow resource allocation problem. Both equations (8) and (9) can be solved by decomposing the problem into M separable, concave resource allocation problems, one for each class in equation (8) and one for each server in equation (9). The optimization problem (8) has additional constraints corresponding to the site to server assignments. The two optimization problems shown in equations (8) and (9) then form the basis for a fixed-point iteration. In particular, initial values are chosen for the variables $f_{i,k}$ and equation (8) is solved using the algorithms described hereafter to obtain the optimal control variables $k_{i,k}^{(j)*}$. This set of optimal control variables $k_{i,k}^{(j)*}$ are then substituted into equation (9) and the optimal control variables $f_{i,k}^{*}$ are obtained. This iterative procedure continues until a difference between the sets of control variables of an iteration and those of the previous iteration is below a predetermined threshold. The optimization problems are defined more precisely as follows.

Optimization Algorithms

There are, in fact, two related resource allocation problems, one a generalization of the other. Solutions to both of these problems are required to complete the analysis. Furthermore,

the solution to the special problem is employed in the solution of the general problem, and thus, both will be described.

The more general problem pertains to a directed network with a single source node and multiple sink nodes. There is a function associated with each sink node. This function is required to be increasing, differentiable, and concave in the net flow into the sink, and the overall objective function is the (separable) sum of these concave functions. The goal is to maximize this objective function. There can be both upper and lower bound constraints on the flows on each directed arc. In the continuous case, the decision variables are real numbers. However, for the discrete case, other versions of this algorithm may be utilized. The problem thus formulated is a network flow resource allocation problem that can be solved quickly due to the resulting constraints being submodular.

Consider a directed network consisting of nodes V and directed arcs A. The arcs a_{v1v2} c A carry flow f_{v1v2} from nodes v_1 c V to nodes v_2 c V. The flow is a real variable which is constrained to be bounded below by a constant i_{v1v2} and above by a constant u_{v1v2}. That is,

$$i_{v1v2} [f_{v1v2} [u_{v1v2} \qquad (10)$$

for each arc a_{v1v2}. It is possible, of course, that $i_{v1v2} = 0$ and $u_{v1v2} = °$. There will be a single source node s c V satisfying $Sa_{v2}f_{sv2} - Sa_{v1}f_{v1s} = R > 0$. This value R, the net outflow from the source, is a constant that represents the amount of resource available to be allocated. There are N sinks v_2 c N ` A which have the property that their net inflow $Sa_{v1v2}f_{v1v2} - Sa_{v2v3}f_{v2v3} > 0$. All other nodes v_2 c A-{s} - N are transhipment nodes that satisfy $Sa_{v1v2}f_{v1v2} - Sa_{v2v3}f_{v2v3} = 0$. There is a single increasing, concave, differentiable function F_{v2} for the net flow into each sink node j. So the overall objective function is

$$\sum_{V2cN} F_{v2}\left(\sum_{av1v2} f_{v1v2} - \sum_{av2v3} f_{v2v3}\right) \qquad (11)$$

which is sought to be maximized subject to the lower and upper bound constraints described in equation (10).

A special case of this problem is to maximize the sum

$$\sum_{V2=1}^{N} (F_{v2}(x_{v2})) \qquad (12)$$

of a separable set of N increasing, concave, differentiable functions subject to bound constraints

$$l_{v2} [x_{v2} [u_{v2} \qquad (13)$$

and subject to the resource constraint

$$\sum_{V2=1}^{N} x_{v2} = R \qquad (14)$$

for real decision variables x_{v2}. In this so-called separable concave resource allocation problem, the optimal solution occurs at the place where the derivatives $F_{v2}'(x_{v2})$ are equal and equation (14) holds, modulo the bound constraints in equation (13).

More precisely, the algorithm proceeds as follows: If either $\sum_{v2=1}^{N} l_{v2} > R$ or $\sum_{v2=1}^{N} u_{v2} < R$, there is no feasible solution and the algorithm terminates. Otherwise, the algorithm consists of an outer bisection loop that determines the value of the derivative D and a set of N inner bisection loops that find the value of $l_{v2} [x_{v2} [u_{v2}$ satisfying $F_{v2}'(x_{v2}) = D$ if $F_{v2}'(l_{v2}) [D$ and $F_{v2}'(u_{v2})$ m D. Otherwise x_{v2} is set to l_{v2} (in the first case), or x_{v2} is set to u_{v2} (in the second case). The initial values for the outer loop can be taken as the minimum of all values $F_{v2}'(l_{v2})$ and the maximum of all values $F_{v2}'(u_{v2})$. The initial values for the u_2-th inner loop can be taken to be l_{v2} and u_{v2}.

The more general problem is itself a special case of the so-called submodular constraint resource allocation problem. It is solved by recursive calls to a subroutine that solves the problem with a slightly revised network and with generalized bound constraints

$$l_{v1v2}' \leq f_{v1v2} \leq u_{v1v2}' \qquad (15)$$

instead of those in equation (10). As the algorithm proceeds it makes calls to the separable concave resource allocation problem solver. More precisely, the separable concave resource allocation problem obtained by ignoring all but the source and sink nodes is solved first. Let x_{v2} denote the solution to that optimization problem.

In the next step, a supersink t is added to the original network, with directed arcs jt from each original sink, forming a revised network (V',A'). L_{jt}' is set to 0 and u_{jt}' is set to x_{v2} for all arcs connecting the original sinks to the supersink. For all other arcs, the lower and upper bounds remain the same. Thus, $l_{v1v2}' = l_{v1v2}$ and $u_{v1v2}' = u_{v1v2}$ for all arcs a_{v1v2}. The so-called maximum flow problem is then solved to find the largest possible flow f_{v1v2} through the network (V',A') subject to constraints in equation (10). A simple routine for the maximum flow problem is the labeling algorithm combined with a path augmentation routine. Using the residual network one can simultaneously obtain the minimum cut partition. For definitions of these terms, please see Ahvja, Magnant, & Orlin, Network Flows, Prentice Hall, Englewood Cliffs, NJ 1993, pages 44-46, 70, 163 and 185, which is hereby incorporated by reference. Those original sink nodes j which appear in the same partition as the supersink are now regarded as saturated. The flow f_{v2t} becomes the lower and upper bounds on that arc. Thus, l_{v2t}' is set to u'_{v2t} which is equal to f_{v2t}. For all remaining unsaturated arcs j, l_{v2t}' is set to x_{v2} and u_{v2t}' is set equal to f_{v2t}. Now the process is repeated, solving the separable concave resource allocation problem for the unsaturated nodes only, with suitably revised total resource, and then solving the revised network flow problem. This process continues until all nodes are saturated, or an infeasable solution is reached.

An example of the network flow model is provided in Figure 7. In addition to the source node s, there are NK nodes corresponding to the sites and classes, followed by two pairs of MK nodes corresponding to the servers and classes, and a supersink t. In the example, $M=N=K=3$. In the first group of arcs, the (j,k)th node has capacity equal to $L_k^{(j)}$. The second group of arcs corresponds to the assignment matrix $A(i,j,k)$, and these arcs have infinite capacity. The capacities of the third group of arcs on (i,k) correspond to the SLA constraints. The duplication

of the nodes here handles the fact that the constraint is really on the server and class nodes. The final group of arcs connects these nodes to the supersink *t*.

SLA Profit Maximization: PPS Case

In formulating the SLA based optimization problem under the PPS discipline for allocating server capacity among the classes of requests assigned to each server, the approach is again to decompose the model to isolate the per-class queues at each server. However, in the PPS case, the decomposition of the per-class performance characteristics for each server *i* is performed in a hierarchical manner such that the analysis of the decomposed model for each class *k* in isolation is based on the solution for the decomposed models of classes 1,....,*k*-1.

Assuming that the lower priority classes do not interfere with the processing of class 1 requests, as is the case under PPS, then the product-form results derived above indicate that the arrival process to the class 1 queue is a Poisson process. Hence, equation (5) still holds for class 1 requests which then leads to the following formulation for the class 1 optimal control problem:

$$\text{Max} \quad \sum_{i=1}^{M} P_k^{+} k_{i,1} - (P_1^{+} + P_1^{-}) k_{i,1} \exp(-(l_{i,1} C_i - k_{i,1}) z_1) \qquad (16)$$

$$\text{s.t. } k_{i,1}[\, ln(a_1 z_1)/z_1 + l_{i,1} f_{i,1} C_i, \; i=1,...,M;$$

$$\sum_{J=1}^{N} k_{i,1}^{(j)} = k_{i,1}, \; i=1,...,M;$$

$$\sum_{i=1}^{M} k_{i,1}^{(j)} = L_1^{(j)}, \; j=1,...,N;$$

$$k_{i,1}^{(j)}=0, \text{ if } A(i,j,1)=0, \; i=1,...,M, \; j=1,...,N;$$

$$k_{i,1}^{(j)} \text{m0, if } A(i,j,1)=1, \; i=1,...,M, \; j=1,...,N;$$

where $k_{i,1}^{(j)}$ are the decision variables and all other variables are as defined above.

Upon solving equation (16) to obtain the optimal control variables $k_{i,1}^{(j)*}$, it is sought to statistically characterize the tail distribution for the class 2 queue which will then be used

(recursively) to formulate and solve the optimization problem for the next class(es) under the PPS ordering. Thus, for any class k equal to 2 or more, it is assumed that there are constants $c_{i,k}$ and $h_{i,k}$ such that

$$P[T_{i,k}>x] \text{ j } c_{i,k}e^{-hi,kx} \quad i=1,...,M,\ k=1,...,K \qquad (17)$$

Assuming that the optimization problem for classes 1,...,k-1 have been solved, the control problem for class k can be formulated as:

$$\text{Max} \quad \sum_{i=1}^{M} P_k^{+}k_{i,k}-(P_k^{+}+P_k^{-})k_{i,k}c_{i,k}exp(-h_{i,k}z_k) \qquad (18)$$

$$k_{i,k}[\ ln(a_kz_k)/z_k + (C_i-Sp_{i,k'})l_{i,k},\ i=1,...,M;$$

$$\sum_{j=1}^{N} k_{i,k}^{(j)} = k_{i,k},\ i=1,...,M;$$

$$\sum_{i=1}^{M} k_{i,k}^{(j)} = L_k^{(j)},\ j=1,...,N;$$

$$k_{i,k}^{(j)}=0, \text{ if } A(i,j,k)=0,\ i=1,...,M,\ j=1,...,N;$$

$$k_{i,k}^{(j)}\text{m0, if } A(i,j,k)=1,\ i=1,...,M,\ j=1,...,N;$$

where $k_{i,k}^{(j)}$ are the decision variables and all other variables are as defined above.

In order to apply the optimization algorithms described above, appropriate parameters for $c_{i,k}$ and $h_{i,k}$ must be selected. In one embodiment, the parameters for $c_{i,k}$ and $h_{i,k}$ are selected based on fitting the parameters with the first two moments of the response time distribution, however, other methods of selecting parameters for $c_{i,k}$ and $h_{i,k}$ may be used without departing from the spirit and scope of the present invention.

It follows from equation (17) that for i=1,...,M, k=1,...,K:

$$ET_{i,k} = c_{i,k}/h_{i,k},\ ET_{i,k}^2 = 2c_{i,k}/h_{i,k}^2 \qquad (19)$$

so that

$$h_{i,k} = 2ET_{i,k}/ET_{i,k},\ c_{i,k} = 2ET_{i,k}^2/ET_{i,k} \qquad (20)$$

where $ET_{i,k}$ and $ET_{i,k}^2$ are the first and second moments of $T_{i,k}$, respectively. Using known formulae for $ET_{i,k}$ and $ET_{i,k}^2$, the equations become:

$$ET_{i,k} = (S^k_{k'=1} k_{i,k} b_{i,k'}^{(2)}/2(1-q_{i,k-1}^+)(1-q_{i,k}^+)) + b_{i,k}/(1-q_{i,k-1}^+) \qquad (21)$$

and

$$ET_{i,k}^2 = (S^k_{k'=1} k_{i,k} b_{i,k'}^{(3)}/3(1-q_{i,k-1}^+)^2(1-q_{i,k}^+)) + b_{i,k}^{(2)}/(1-q_{i,k-1}^+)^2 + ((S^k_{k'=1} k_{i,k} b_{i,k'}^{(2)}/(1-q_{i,k-1}^+)(1-q_{i,k}^+)) + S^{k-1}_{k'=1} k_{i,k} b_{i,k'}^{(2)}/(1-q_{i,k-1}^+)^2)ET_{i,k} \qquad (22)$$

where $b_{i,k'}, b_{i,k'}^{(2)} b_{i,k'}^{(3)}$ are the first three moments of the service times, $q_{i,k}^+$ is the total load of classes 1,...,k:

$$q_{i,k}^+ = S^k_{k'=1}\ q_{i,k'}\ \mathrm{h}\ S^k_{k'=1} k_{i,k} b_{i,k}/C_i \qquad (23)$$

When the service requirements can be expressed as mixtures of exponential distributions, the cost functions of equation (18) are concave. Therefore, the network flow model algorithms can be recursively applied to classes 1,2,...,K.

SLA Profit Maximization: General Workload Model

The optimization approach described above can be used to handle the case of even more general workload models in which the exponential exogenous arrival and service process assumptions described above are relaxed. As in the previous cases, analytical expressions for the

tail distributions of the response times are derived. In the general workload model, the theory of large deviations is used to compute the desired functions and their upper bounds.

Consider a queuing network composed of independent parallel queues, as shown in **Figures 6A** and **6B**. The workload model is set to stochastic processes $U_k^{(j)}(t)$ representing the amount of class k work destined for site j that has arrived during the time interval $(0,t)$. The workload model defined in this manner corresponds to Web traffic streams at the request level rather than at the session level, which was the case described above with regard to **Figures 6A** and **6B**.

Let $b_{i,k}$ be the decision variables representing the proportion of traffic of $U_k^{(j)}$ to be sent to server i: $U_{i,k}^{(j)}(t) = b_{i,k}A(i,j,k)U_k^{(j)}$. Let $V_{i,k}(t)$ be the potential traffic of class k set to server i during the time interval $(0,t)$:

$$V_{i,k}(t) = S^{N}_{j=1} A(i,j,k)U_k^{(j)} \qquad (25)$$

and define

$$q_{i,k} = \lim_{t\to\infty} (1/t)V_{i,k}(t) \qquad (26)$$

to be associated asymptotic potential load of class k at server i, provided the limit exists. Further let

$$U_{i,k}(t) = S^{N}_{j=1} U_{i,k}^{(j)}(t) = S^{N}_{j=1} b_{i,k}A(i,j,k)U_k^{(j)}b_{i,k}V_{i,k}(t) \qquad (27)$$

denote the class k traffic that has been sent to server i during the time interval $(0,t)$. Thus,

$$\lim_{t\to\infty} (1/t)U_{i,k}(t) = b_{i,k}q_{i,k} \qquad (28)$$

Assume again that GPS is in place for all servers with the capacity sharing represented by the decision variables $f_{i,k}$. The SLA under consideration is

$$P[W_k>z_k][a_k, k=1,...,K-1 \qquad (29)$$

where W_k is the remaining work of class k at any server.

Bounds of the tail distributions of the remaining class k work at server i are considered by analyzing each of the queues in isolation with service capacity $f_{i,k}C_i$ (see **Figure 6B**). Such bounds exist for both arbitrary and Markovian cases. For tractability of the problem, it is assumed that $b_{i,k}q_{i,k}<f_{i,k}C_i$.

Only asymptotic tail distributions given by the theory of large deviations are considered:

$$P[W_{i,k}>z_k] \text{ i } \exp(-h_{i,k}z_k) \qquad (30)$$

where $W_{i,k}$ is the remaining work of class k at server I. In order to apply the large deviations principal, it is assumed that for all $i=1,...,M$ and $k=1,...,K$, the following assumptions hold:

(A1) the arrival process $V_{i,k}(t)$ is stationary and ergodic (see S. Karlin et al., A First Course in Stochastic Processes, 2nd Ed., Academic Press, San Diego, CA, 1975, pages 443 and 487-488, which is hereby incorporated by reference); and

(A2) for all $0<h<^{\circ}$, the limit $L_{i,k}(h)=\lim_{t\to^{\circ}}(1/t)\log_e exp(hV_{i,k}(t))$

exists, and $L_{i,k}(h)$ is strictly convex and differentiable.

Note that for some arrival processes, assumption (A2) is valid only through a change of scaling factor. In this case, the asymptotic expression of the tail distribution of the form in equation (30) could still hold, but with a subexponential distribution instead of an exponential one.

It then follows that under assumptions A1 and A2, the arrival processes Vi,k(t) satisfy the large deviations principal with the rate function

$$L_{i,k}(a)^* = \sup(ha - L_{i,k}(h)) \qquad (31)$$

h

where $L_{i,k}(a)^*$ is the Legendre transform of $L_{i,k}(h)$.

Now let

$$L_{i,k}(h,b_{i,k}) = \lim_{t\to\infty} (1/t)\log E e^{hU_{i,k}(t)}$$

$$= \lim_{t\to\infty} (1/t)\log E e^{h\, b_{i,k} V_{i,k}(t)}$$

Then, $L_{i,k}(h,b_{i,k}) = L_{i,k}(hb_{i,k})$, and thus, the exponential decay rate $h_{i,k}^*$ is defined by

$$h_{i,k}^* = \sup\{h : L_{i,k}(hb_{i,k})/h < f_{i,k}C_i\} \qquad (32)$$

This exponential decay rate is a function of $C_{i,k}$ h $f_{i,k}C_i$ and $b_{i,k}$, which will be denoted by $h_{i,k}^*(b_{i,k},C_{i,k})$. As $h_{i,k}^*(b_{i,k},C_{i,k})$ decreases and is differentiable in b, $x_{i,k}(C,h)$ can be defined as the inverse of $h_{i,k}^*(b_{i,k},C_{i,k})$ with respect to b, i.e. $x_{i,k}(C,h_{i,k}^*(b_{i,k},C_{i,k})) = b$. The corresponding optimization problem may then be formulated as:

$$\text{Max} \quad \sum_{i=1}^{M}\sum_{k=1}^{K-1} P_k^+ b_{i,k} p_{i,k} - (P_k^+ + P_k^-) b_{i,k} \exp(-h_{i,k} z_k) \qquad (33)$$

$$\text{s.t. } b_{i,k}[\, x_{i,k}(f_{i,k}C_i, -\log(a_k z_k)/z_k),\ i=1,\ldots,M, \quad k=1,\ldots,K-1; \qquad (34)$$

$$b_{i,k} < f_{i,k}C_i/p_{i,k},\ i=1,\ldots,M, k=1,\ldots,K-1;$$

$$\sum_{i=1}^{M} b_{i,k} = 1,\ k=1,\ldots,K-1;$$

$$\sum_{k=1}^{K-1} f_{i,k}\,[1,\ i=1,\ldots,M.$$

Note that the constraint in equation (34) comes from the relaxed SLA requirement $h_{i,k}^*$m-$\log(a_k z_k)/z_k$.

Owing to the above, the function $b\exp(-zh_{i,k}^*(b,C))$ is convex in b, so that the cost function is also concave in the decision variables $b_{i,k}$. Thus, the optimization alogrithms described above may be iteratively used to solve the control problem.

In the case of Markovian traffic models, such as Markov Additive Processes and Markovian Arrival Processes (see D. Lucantun et al., "A single Server Queue with Server Vacations and a Class of Non-renewal Arrival", Advances in Applied Probability, vol. 22, 1990, pages 676-705, which is hereby incorporated by reference), such functions can be expressed as the Perron-Ferobenius eigenvalue. Thus, efficient numerical and symbolic computational schemes are available.

Figure 8 is a flowchart outlining an exemplary operation of the present invention when optimizing resource allocation of a Web server farm. As shown in **Figure 8**, the operation starts with a request for optimizing resource allocation being received (step **810**). In response to receiving the request for optimization, parameters regarding the Web server farm are provided to the models described above and an optimum solution is generated using these models (step **820**). Thereafter, the allocation of resources is modified to reflect the optimum solution generated (step **830**). This may include performing incremental changes in resource allocation in an iterative manner until measure resource allocation metrics meet the optimum solution or are within a tolerance of the optimum solution.

Figure 9 provides a flowchart outlining an exemplary operation of a resource allocation optimizer in accordance with the present invention. The particular resource allocation optimizer shown in **Figure 9** is for the GPS case. A similar operation for a PPS resource allocation optimizer may be utilized with the present invention as discussed above.

As shown in **Figure 9**, the problem of finding the optimal arrival rates $k_{i,j}^{(k)}$ for the two best effort class are solved (step **910**). This is outlined in equation (6) above. Next, the GPS parameters $f_{i,k}$ for each server i and class $k<K$ are initialized (step **920**). The value of initial parameters may be arbitrarily selected or may be based on empirical data indicating values providing rapid convergence to an optimal solution.

The "previous" arrival rate and GPS parameters $k_{i,j}^{(k)}$ (old) and $f_{i,k}$ (old) are initialized (step **930**). The choice of initialization values forces the first test to determine convergence of $k_{i,j}^{(k)}$ and $f_{i,k}$ to fail. The class is initialized to $K=1$ (step **940**) and the problem outlined in equation (8) is solved to obtain the optimal values of the arrival rates $k_{i,j}^{(k)}$ given the values of the GPS parameters $f_{i,j}$ (step **950**).

The class k is then incremented (step **960**) and it is determined whether $k<K-1$ (step **970**). If it is, the operation returns to step **950**. Otherwise, the server is initialized to $i=1$ (step **980**). The problem outlined in equation (9) is solved to obtain the optimal values of the GPS parameters $f_{i,j}$ given the values of the arrival rates $k_{i,j}^{(k)}$ (step **990**).

The server i is then incremented (step **1000**) and a determination is made as to whether $i<M$ (step **1010**). If it is, the operation returns to step **990**. Otherwise, convergence of the $k_{i,j}^{(k)}$ values is checked by comparing them with the $k_{i,j}^{(k)}$ (old) values (step **1020**). If there is no convergence, the each old arrival rate valve $k_{i,j}^{(k)}$ (old) is reset to be $k_{i,j}^{(k)}$, and each old GPS parameter $f_{i,k}$ (old) is reset to be $f_{i,k}$ (step **1030**). The operation then returns to step **940.**

If there is convergence in step **1020**, convergence of the $f_{i,k}$ values is checked by comparing them with the $f_{i,k}$ (old) values (step **1040**). If there is no convergence the operation goes to step **1030**, described above. Otherwise the optimal arrival rates $k_{i,j}^{(k)}$ and the optimal GPS parameters $f_{i,k}$ for each server i, site j and class K have been identified and the operation terminates.

Thus, the present invention provides a mechanism by which the optimum resource allocation may be determined in order to maximize the profit generated by the computing system. The present invention provides a mechanism for finding the optimal solution by modeling the computing system based on the premise that revenue is generated when service level agreements are met and a penalty is paid when service level agreements are not met. Thus, the present invention performs optimum resource allocation using a revenue metric rather than performance metrics.

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media, such as a floppy disk, a hard disk drive, a RAM, CD-ROMs, DVD-ROMs, and transmission-type media, such as digital and analog communications links, wired or wireless communications links using transmission forms, such as, for example, radio frequency and light wave transmissions. The

computer readable media may take the form of coded formats that are decoded for actual use in a particular data processing system.

The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiments were chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.